DTIC FILE COPY

GL-TR-89-0286

# Organization, Access, and Exploration Facilities for Large Geophysical Databases: DATSAV Surface Meteorological Data

AD-A225 132


Rudolf B. Husar

Center for Air Pollution Impact and Trend Analysis
Washington University
St. Louis, MO 63130




19 September 1989

Scientific Report #1

Approved for public release; distribution unlimited


GEOPHYSICS LABORATORY
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
HANSCOM AIR FORCE BASE, MA 01731-5000

"This technical report has been reviewed and approved for publication"

(Signature)
Eric P. Shettle
Contract Manager

(Signature) for
Donald E. Bedo, Chief
Atmospheric Optics Branch

FOR THE COMMANDER

(Signature)
R. Earl Good, Director
Optical and Infrared Technology Division

This report has been reviewed by the ESD Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS).

Qualified requestors may obtain additional copies from the Defense Technical Information Center. All others should apply to the National Technical Information Service.

If your address has changed, or if you wish to be removed from the mailing list, or if the addressee is no longer employed by your organization, please notify GL/IMA, Hanscom AFB, MA 01731. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be turned.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

**1a. REPORT SECURITY CLASSIFICATION**

**1b. RESTRICTIVE MARKINGS**

**2a. SECURITY CLASSIFICATION AUTHORITY**
Unclassified

**2b. DECLASSIFICATION/DOWNGRADING SCHEDULE**

**3. DISTRIBUTION/AVAILABILITY OF REPORT**
Approved for public release;
distribution unlimited

**4. PERFORMING ORGANIZATION REPORT NUMBER(S)**
Washington University

**5. MONITORING ORGANIZATION REPORT NUMBER(S)**
GL-TR-89-0286

**6a. NAME OF PERFORMING ORGANIZATION**
Washington University

**6b. OFFICE SYMBOL** *(If applicable)*

**7a. NAME OF MONITORING ORGANIZATION**
Geophysics Laboratory

**6c. ADDRESS** *(City, State, and ZIP Code)*
Center for Air Pollution Impact & Trend Analys
Campus Box 1124
St. Louis, MO 63130

**7b. ADDRESS** *(City, State, and ZIP Code)*
Hanscom Air Force Base, MA 01731-5000

**8a. NAME OF FUNDING/SPONSORING ORGANIZATION**

**8b. OFFICE SYMBOL** *(If applicable)*

**9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER**
F19628-87-K-0003

**8c. ADDRESS** *(City, State, and ZIP Code)*

**10. SOURCE OF FUNDING NUMBERS**

| PROGRAM ELEMENT NO. | PROJECT NO. | TASK NO. | WORK UNIT ACCESSION NO. |
|---|---|---|---|
| 62101F | 7670 | 15 | AO |

**11. TITLE** *(Include Security Classification)*
Organization, Access, and Exploration Facilities for Large Geophysical Database:
DATSAV Surface Meteorological Data

**12. PERSONAL AUTHOR(S)**
Rudolf B. Husar

**13a. TYPE OF REPORT**
Scientific 1

**13b. TIME COVERED**
FROM 88/10/01 TO 89/09/19

**14. DATE OF REPORT** *(Year, Month, Day)*
89/09/19

**15. PAGE COUNT**
44

**16. SUPPLEMENTARY NOTATION**

**17. COSATI CODES**

| FIELD | GROUP | SUB-GROUP |
|---|---|---|
| | | |
| | | |

**18. SUBJECT TERMS** *(Continue on reverse if necessary and identify by block number)*
Visibility Database Management (EDC)

**19. ABSTRACT** *(Continue on reverse if necessary and identify by block number)*


The overall objective of this project is to derive atmospheric aerosol climatology for the continental areas of the world. This is to be achieved by combining archived meteorological data observations, aerosol monitoring data, and general knowledge on the properties and behavior of aerosols. Exploration of meteorological data is a data intensive project. In this report database concepts and available data handling and manipulation techniques are described. Novel data handling and exploration facilities were applied to the meteorological data. Rationale and specific methods used analyzing European visibility data are presented. Keywords:


**20. DISTRIBUTION/AVAILABILITY OF ABSTRACT**
☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

**21. ABSTRACT SECURITY CLASSIFICATION**
Unclassified

**22a. NAME OF RESPONSIBLE INDIVIDUAL**
Eric P. Shettle

**22b. TELEPHONE** *(Include Area Code)*
(617) 377-3665

**22c. OFFICE SYMBOL**
GL/OPA





A

CONTENTS

## LIST OF FIGURES



## LIST OF TABLES

## ACKNOWLEDGEMENTS

The processing of the DATSAV database was a three year effort during which many individuals contributed. The initial data processing was conducted by Feliks Golenko. Some of the data processing routines were prepared by Masuri Hashim. Their insight and help is gratefully acknowledged.

# 1. INTRODUCTION

## 1.1 Objectives of the Aerosol Climatology Project

The overall objective of the present project is to derive an atmospheric aerosol climatology for the continental areas of the world. This is to be achieved by combining archived meteorological observations, special aerosol monitoring data, and general knowledge on the properties and behavior of aerosols.

The image and other signal deterioration by hazy atmosphere constitutes one of the limiting factors in the performance of many sophisticated sensors, including the human eye. By establishing a global climatology of atmospheric haze, the potential performance of remote sensing devices can be evaluated for most locations in the world. The knowledge of haze characteristics can also be useful as input for design specifications of new sensors. Beyond the applications in the Department of Defense (DoD) a haze climatological database can be used in research on climate change, biogeochemical cycles, and air quality.

The specific goal of this project on Global Aerosol Climate is to provide improved aerosol input into the radiative transfer models LOWTRAN and FASTCODE. These models are used operationally within AF and their performance can be significantly improved by an aerosol climatological database.

## 1.2 DATSAV Database

The main database used in this study is the DATSAV[1] database which consists of world wide surface weather observations collected through the USAF Automated Weather Network (AWN). It is decoded at the Air Force Global Weather Central (AFGWC), Offutt AFB, NE, and stored on magnetic tape at USAFETAC, Scott AFB, IL, and operating location A, Asheville, NC.

DATSAV contains worldwide surface observations for about 13,000 stations. The stored variables include:

- Wind direction
- Wind speed
- Barometric pressure
- Pressure tendency and change
- Dry bulb temperature
- Dew Point temperature
- Total sky cover
- Visibility
- Past and present weather
- Cloud layer data
- Ceiling
- Precipitation data
- Snowfall and snow depth
- Runway data
- Hail data
- Sunshine data
- Ground temperature and conditions
- Maximum and minimum temperatures
- Ship data
- Sea surface temperature
- Wave data
- Swell data
- Ship ice report

The period of record is from January 1973 to 1986. The data file size is 56 tapes/yr ( one tape for a station file). Most data elements checked extensively for transmission and decode errors; errors corrected using systematic algorithms. Station files undergo additional quality control through manual interaction.

The subset of the DATSAV database used in this project includes all Europe and outlying areas such as Greenland. In this geographic domain data for 1,600 stations are available. The size of the total database is about 30 gigabytes.

## 1.3 Objectives of this Report

In order to create an aerosol climate data base, most of the effort in this project has been directed toward accessing, performing quality control, manipulating, and exploring large quantities of environmental data, mostly from meteorological data archives. The aerosol science aspect of this project was less demanding.

In much of the past environmental data analysis, the methods of data organization and handling were considered of secondary importance. Data manipulation was the activity of "technicians", such as computer programmers. Also, the data handling or "engineering" was conducted *ad hoc* using the methods, tools, and skills of the investigator.

There are compelling reasons to improve the data handling and exploration facilities applied in environmental-meteorological research particularly in data intensive projects such as the current project. The possible benefits include efficiency of data access, improved quality, and enhanced quality of the extracted knowledge.

During the 1980's there were significant advances in database technology, user interface design, and computer hardware. At the same time the quantity of machine readable environmental data sets has increased significantly. These developments warrant a deeper examination and evaluation of the data handling and exploration practices.

The motivation for this report was the need to describe the data organization and handling concepts used for the present Aerosol Climatology Project. However, the rationale as well as the specific methods developed/reviewed here have applications to other environmental data intensive projects within and outside the Department of Defense.

### 1.4 Organization of this Report

The body of this report has four major sections. Section two is a statement of purpose of the data processing i.e. producing higher grade knowledge from raw data. In particular the major resistances in knowledge (report, publication) production are identified and possible remedies discussed. Section three focuses on data organization methodologies, in the past, present and future. Special emphasis is given to the application of the relational data model to spatial and temporal data. Section four deals with data manipulation approaches. Interactive graphic tools for data quality control, exploration and presentation are considered in detail. Throughout the report, the specific examples were taken from the Aerosol Climatology Project.

## 2. PROCESSING OF DATA INTO KNOWLEDGE

The ultimate purpose of most data collections and analysis is to produce new knowledge in form of a report or published paper. Scientific knowledge is created by first observing nature, noting patterns and relationships and then assembling a theory (model) to explain the observations.

Through computers and telecommunication, the behavior of the real word can be recorded, stored in data bases and replayed, analyzed and simulated through data display software or mathematical models. The replaying of real world observations and simulation (modeling) can help us obtain experience artificially. Acquisition of knowledge through "artificial experience" i.e. data exploration, analysis, may be significantly faster than first-hand experience.

Knowledge is used mainly for decision making. In some cases the generated knowledge can be used directly. In other cases the generated knowledge is simply a piece in a larger system or conceptual model that ties other pieces of physical knowledge together. As an example, one piece of declarative knowledge (fact) is the aerosol optical depth at a specific location on the globe and specific season. The optical depth *per se* is of marginal utility. It is seldom used for decision making. However, when incorporated in a global radiation model or in a radiative transmission model, then it can contribute to decision making.

Knowledge acquisition may be viewed as a process:

| | | | |
|---|---|---|---|
| DATA -> | INFORMATION -> | KNOWLEDGE -> | WISDOM |
| Datum point | Data base | Publication | Experience |

The process is like a refinery or manufacturing plant: raw data and old knowledge in refined, new knowledge out.

Producing knowledge from data i.e. scientific data analysis can take many forms and it is largely at an investigator's discretion to choose the most appropriate pathway. It is a process that is driven largely by intuitive decisions, and it is highly dependent on feedback; a decision of what to do next depends on the outcome of previous steps. Automating this process of data "refining" into usable knowledge through artificial intelligence i.e. computer logic is not feasible or meaningful at this time. Nevertheless, there are significant advances in AI in that direction (e.g. Parsaye and Chignell, 1983)(2) It is much more realistic to seek an improvement of the efficiency of the human investigator in performing the data manipulation and exploration tasks, i.e. augmenting human intelligence.

## 2.1 Types of Environmental Knowledge

The refinement of data into knowledge requires a specification of the vague term "knowledge". For environmental systems it is convenient to categorize knowledge into descriptive, process orientated and explanatory knowledge as illustrated in Figure 1.

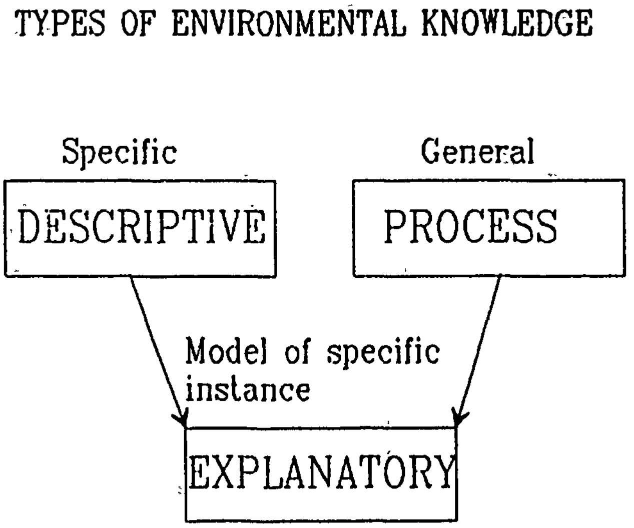


**Figure 1. Types of environmental knowledge.**

**Descriptive knowledge** This type of knowledge is composed of facts, data items that characterize the environmental variables at specific locations, and time. It is determined directly from raw data or by filtering, extracting, or otherwise summarizing the observations. This type of knowledge can be extracted from the data without the need to invoke deeper knowledge such as physical processes and operating mechanisms. The applicability of this knowledge is limited to a particular instance of space/time.

Extracting descriptive knowledge requires standard statistical tools for data processing. It is therefore amenable to automation using the computer. A typical example is climatology. It does not "explain", merely states the main, long term characteristics of weather.

**Process oriented knowledge.** Textbooks or tutorials typically explain the "theory" of a phenomenon including governing processes, equations, and solution methods. This type of knowledge is more abstract then general descriptive knowledge. It is applicable to a broad domain of instances.

Extracting process orientated information is in fact, devising a new theory for a process. It requires deep knowledge and the unique features of human intelligence. It is not reachable through machine intelligence. A typical example is the theory of light scattering in terms of the general conservation equations.

**Explanatory Knowledge.** This is the highest grade of knowledge, in that, it can explain the observations in terms of an applicable theory. It requires knowledge of the facts for a specific instance and the application of the deeper theory that explains it.

Typical examples of explanatory knowledge are: realistic simulations of a specific thunderstorm, weather forecasting, a radiation model reliably simulating the light transmission through the atmosphere.

It is evident that computer data processing and simulation is geared toward improving the acquisition and quality of descriptive and explanatory knowledge. Discovery of new theories requires the unique reasoning and associative power of human mind. Creation of process oriented knowledge, i.e. new theories will be the domain of human intelligence for a some time.

## 2.2 Characteristics of Environmental Data Sets

Before considering the data exploration and knowledge extraction processes it is helpful to state the main characteristics of environmental data sets: their domain, size, and texture.

**Domains and dimensionality.** The domain of the environmental data extends over multiple dimensions which include time, space, and variables. The time domain can span from seconds to centuries or longer. However, meteorological data sets have typical domains of years with time resolutions of hours or days. Usually the time units between consecutive data points are fixed for a data base.

The spatial domain of environmental data may extend over two (latitude, longitude) or three (height) dimensions. The domain can have a characteristic size of meters to global dimensions ($10^7$ meters). The spatial resolution of data is typically in meters or kilometers.

The parameter of a variable domain of environmental data extends from one or two variables for simple weather station to several hundred variables in a chemical database. A typical environmental database has tens of variables.

In scientific notation the dimensionality of the data space may be expressed as *value = f (space, time, variable) = g(x, y, z, t, i,)*, where *f*(...) represents function of... This means that the data space covers four or five independent dimensions. A further data dimension is introduced when the value of the variable is augmented by other attributes such as error value, quality flag, etc. It is to be noted that the dimensionality of the data set will influence the number of "relations" in a relational data base. This is discussed in detail in section three on database design.

**Database size.** Environmental databases generally qualify as large or very large databases. The multiple dimensions along with the number of discrete points in each dimension may yield many times $10^4$ - $10^8$ data points. Satellite measuring platforms produce the largest known databases containing over $10^{14}$ observations. The sheer size of such databases creates challenges for quality control, storage, access, and dissemination.

The input data to the aerosol climatology project for Europe alone occupied over 500 magnetic tapes or $3*10^{10}$ bytes. The compacted data size is about a tenth of that.

**Data texture.** Most environmental data sets are inherently "noisy" in space/time domain. Much of this "noise" is natural variability and not measurement noise. The data variables have significant spatial and temporal variation. The "signal" is usually obscured by significant amount of noise and data pattern cannot be readily revealed without significant noise reduction processing such as temporal and spatial averaging (filtering). The phenomenon is illustrated using extinction coefficient data for Frankfurt, West Germany, Figure 2.

It is evident that environmental databases posses significant unique features that require special consideration in the design of modern data handling facilities. Nevertheless, it is vital to use existing standard database technology as much as possible.

## 2.3 Data Life Cycle

The life cycle of data during scientific analysis has at least three major phases:

Data acquisition
Data exploration, analysis
Data presentation, dissemination.

The gathering phase involves the acquisition of data from internal to external sources. The analysis phase constitutes the manipulation of raw data for the creation of higher grade information. The presentation phase is the extraction and formatting of analyzed data into graphical, textual, or verbal reports.

In the gathering phase one needs to access and verify the data. This quality assurance and quality control constitutes the main activity in the gathering phase.

The analysis phase may involve routine data extraction and summary or detailed conceptual or numerical modelling using the available data as inputs. In either case, the analysis phase is accomplished by manipulation of existing data, and browsing to establish patterns and relationships.

The presentation of the data is accomplished by transforming the analyzed and summarized data in a form that is most efficient in conveying the substance of the analysis. Maps, charts, tables, and descriptive text are the most common forms of data presentation.

All three phases of the data flow involve numerous activities and problems. One way to improve data exploration performance is to identify and minimize the main resistance that one encounters in the process.

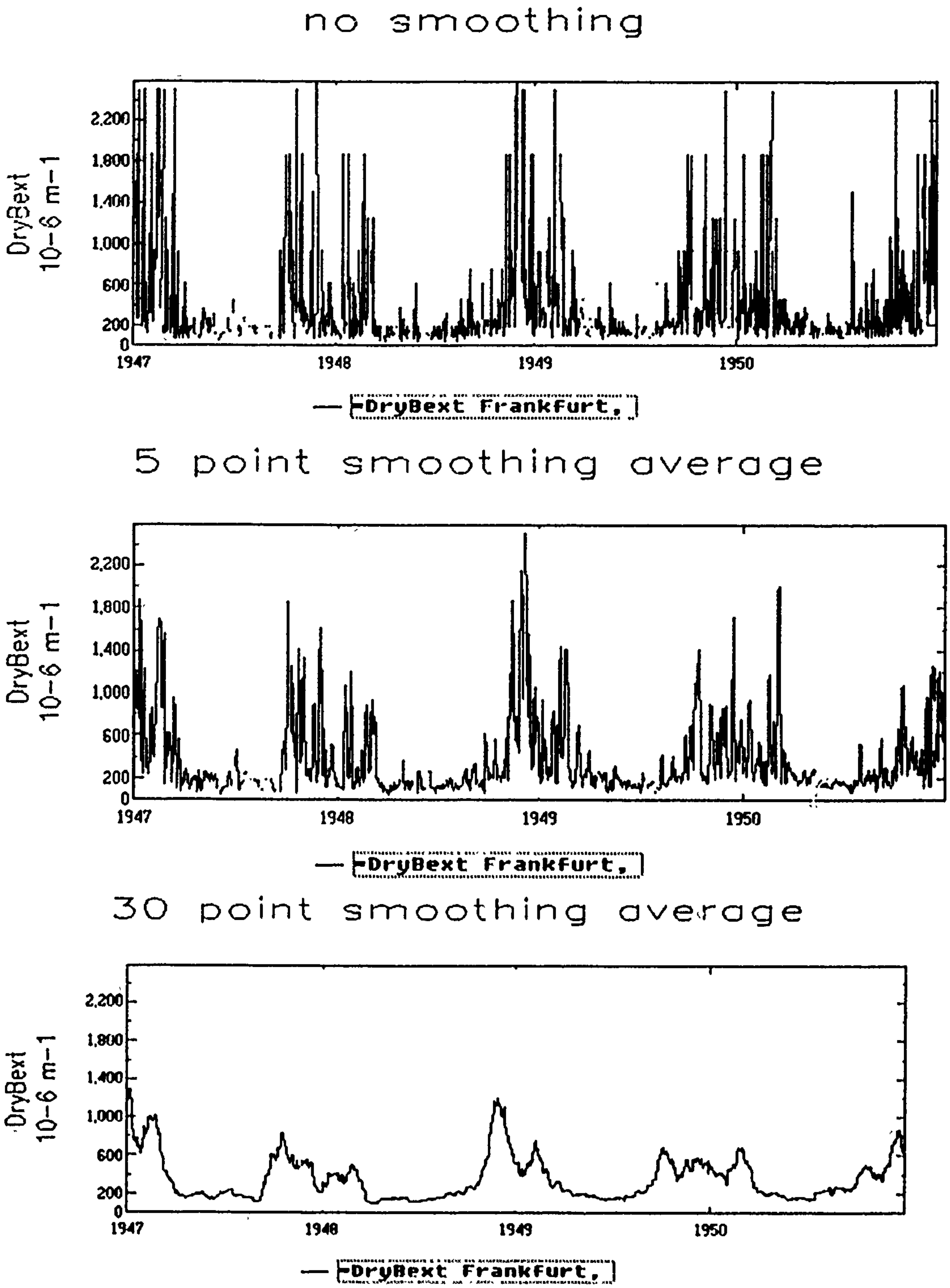


**Figure 2. Time averaging of extinction coefficient.**

### 2.4 Resistances in Knowledge Creation

Data processing into knowledge has numerous resistances. Some are "mechanical" and lend themselves to machine assistance others are conceptual and require new ideas.

**Data Format.** Most current data bases have evolved rather than created by a grand design. Consequently, their purpose and format is extremely heterogeneous. These are like small archipelagos, each having its own peculiarities in form and substance. Accessing such data is impeded by the lack of uniformity and structure. This resistance can be overcome either by standardized data formats or flexible and smart mediators between sources and users of data.

In case of the DATSAV, the data set is contained in a special binary format using random record length. The data are only accessible through special software. Significant effort was required to prepare such software.

**Data are in wrong form.** The proliferation of information sources and quantity may lead one to conclude that the problem is too much data. Actually, humans can absorb, filter, and use immense quantities of data provided that it is in the right form. A good example is the windshield in front of the driver of a car: in computer jargon it is a high resolution color screen with great deal of texture, that changes every 1/30 of a second. In terms of shear bit flow it is giga-bits per second but our eye-brain system has no problems dealing with it: we filter it, analyze it and make instant decisions based on its changing features. The reason, that we can deal with it is because the information is coming in a visual form that we can absorb and analyze of a very high bandwidth. If the information comes as characters on a page of a book or screen, the effective bandwidth of communication is reduced by several orders of magnitude. This is the origin of the saying that a picture is thousand words.

In future information systems, special attention will needed to be given to the proper "packaging" and information delivery system to the human user. This generally involves the use of graphics and sound that provide not only the data but an audio/visual context that gives frames of reference for interpretation. Presenting data and knowledge in terms of a moving picture highlighting user defined significant features, could improve the data assimilation rate dramatically at the same time reducing the burden to the user.

**Knowledge about data is separated from data sets.** Numeric and fielded data contained in records (tuples) contain very little information without a context or explanation. This context is contained in data dictionaries, data description reports and human expertise. Until the 1960's information resources were contained in transaction records, paper forms, telephone and verbal exchanges, etc. There was usually a person who was intimately familiar with the location and meaning of hard copy and loose records as well as its history. That person could exercise significant amount of expert judgement in using the data because he was aware of its form and substance. In the 1960's and 70's with the introduction of computerized data handling system the data and the knowledge about the data proceeded along separate paths. The data were maintained by data processing professionals while the meaning was interpreted by professional data analysts. Commonly, the analysts had little knowledge or insight about the sources of the data, methods of collection, uncertainty, etc. Most current data bases contain only minimal information that describe their meaning. The knowledge on the sources of the data uncertainty and deeper meaning generally resides with human experts and it is not coded along with the data. The users of such data bases have difficulty interpreting such alphanumeric data. This impedes the creation of quality knowledge from data.

The DATSAV database is a good example for inadequate meta data. The meteorological data collected over the world are reformatted in common format for the sake of comparability. However, in the process key pieces of information were lost that hinder the analysis. In the case of visibility data from Soviet Union, for example, many stations show the same visual range over time. However, the user of such data has no access for the reasons as to why this anomaly exists.

Since the late 1980's there is a strong movement to reunite data and the associated knowledge. The concepts and systems that accomplish that are referred to as expert data base systems (e.g. Kerschberg, 1986[3]; Ullman 1988[4]; Brodie and Mylopoulos, 1986[5]) and intelligent databases (Parsaye et al., 1989[6]).

## 3. DATABASE DESIGN

Designing a database requires the understanding of the origin, meaning of the data, implementation issues, such as efficiency, the limitations of coding and accessing data. This process relies heavily on the experience and intuition of the designer. The theoretical guidance for "good" design is minimal. Consequently, there is an immense diversity of data sets that are maintained at research organizations. The trend in the evolution of data organization is illustrated in the Table 1.

Table 1. Evolution of data organization and access methods.

| FILES | COMM FORMAT | DATABASES | KNOWLEDGE BASES |
|---|---|---|---|
| Misc. Format | Common Format | Common Format | Common Format |
| Tailored software. | Common functions | SQL manipulation | SQL manipulation |
| C, FORTRAN, BASIC | C or FORTRAN | Relational model | Hypertext |
| | Some metadata | Some metadata | More metadata |
| | | Integrity constraint | Integrity constraint |
| | | | Rules,Frames,Inferencing |

Most scientific research organizations are still in the single file management phase, with the more advanced places started implementing a common data format for their files. The DATSAV data set falls in the category of a common data format

The discussion below will review the common data formats used in environmental sciences and illustrates the application of the relational data model to organize and access environmental data.

### 3.1 Data Structures for Scientific Databases

In recent years significant amount research was conducted at various scientific organizations on the development of data formats that are not the standard common database models.

Scientific databases have been imposing severe demands on the organizational system for databases. In most instances, scientist were forced to devise their own data organization and manipulation schemes. Several well defined scientific data formats have been developed since the mid 1970's. Examples are the block data set (BDS), (McPherron 1976)(7), the universal format for meteorological radar data (Barnes 1980)(8), FLATDBMS of Smith and Clauer (1986)(9), the common cartesian format

CCF of Mohr et al. (1986)[10], the common data format (CDF) of Treinish and Gough (1987)[11], the common data format of Raymond (1988)[12], and the netCDF data format developed for the NCAR UNIDATA system (Rew, 1988)[13].

The common characteristics of these attempts of defining a data format is that they

1. Accommodate the multidimensionality of environmental data sets.
2. Incorporate some description of the data set (metadata).
3. Use abstract data types.

When expressed through abstract data types the data objects are accessed through a set of routines rather than directly from the users program. This allows a separation of physical data access implementation from the logical data structure (see Appendix A). Also, data can be located on different computer systems having different operating environments. Another feature of these modern data structures is that the logical view, such as a time series or spatial plot of the data is presented to the user in a way that is consistent with the meaning of the data. Earlier data structures were designed to satisfy the needs of the data processor, not the user.

Modern scientific data formats are self-describing. This feature is necessary since the actual form of the multidimensional data differs from one instance to another. Such data formats typically consist of a descriptor section that contains the dimensions of the data, location, time, units, number of variables, etc. This information is usually contained in a header that is attached to the main data table containing the alphanumeric data sets. These headers are generally referred to as data dictionaries or metadata (data about data). A scientific data set may be composed of significant number of different data objects (e.g. tables) each of which have to be self-describing. An extreme case for such a self-describing data structure is a spreadsheet file where every cell is an object that describes itself and its relationship to other cells. It should be noted that the DATSAV data format is common only to the DATSAV database and that it is not self describing. The DATSAV data dictionary is contained in a separate hard copy report.

There is a direct relationship between the complexity of the data structures and the complexity of the software that needs the access in manipulate such data. It is therefore, desirable to strike an optimal data structure complexity that accommodates the need of the data set but not overly complex to impede processing.

Recently introduced data formats for scientific data are more general than their predecessors. Nevertheless, they require special software that deals explicitly with the access in manipulation of data in that particular format. The processing of the data is performed in "third generation languages", such as FORTRAN, C, BASIC, or PASCAL. The use of standard commercial software for the management of that scientific data has been limited. Among the reasons for not utilizing such database management systems, (DBMS), include (Rew, 1988[13]):

1. Existing DBMS's have poor support for the complex data structures required for scientific data.

2. General-purpose database systems show poor performance on large scientific data sets. Satellite images, scientific model outputs, weather observations covering several centuries are generally beyond the capabilities of most DBMS to organize and efficiently access such data.

3. Many of the facilities of general purpose database management systems are unnecessary for scientific data, while other facilities such as array manipulation and heavy numerical processing are not part of such systems.

Consequently most of the scientific database handling systems use data organization and access methods that are tailored to their own needs. It is believed that limitations of existing standard DBMS will be overcome in the near future.

As noted earlier the DATSAV database can only be accessed and manipulated through special software and standard data management software is unsuitable to access DATSAV, because of its unconventional data format and large size.

### 3.2 Modern Data Flow Handling Concepts

Scientific databases frequently require special preparation, processing, and display that is not feasibly through standard database management systems. Such customized processing can be aided significantly by considering some of the recent conceptual developments in data handling.

**Data flow.** Future data manipulation components are envisioned as modules placed along the *data highway.* Such modules may include physical data access, data manipulation, display, and other modules. These specific modules can be handed to specialized hardware or software for efficient execution.

The *data highway* is best envisioned as flow diagrams, where data are passed from one module to another (e.g. UNIX modules). A key role is assigned to a mediator module that handles the data transfer between the user application and data sources. Modules should have multiple pathways in and out of each module and data should include graphic objects.

The linkage of components in such a system is established by the user by selecting individual tools and connecting them in order to establish a data flow diagram. An example toolbox developed for atmospheric data manipulation was reported by Raymond (1988[11]). In advanced data flow systems the user may change the controls within one process and observe the changes in the other windows (processes) (e.g. Haeberli 1988, 1989)[14,15]. This allows very high, process level programming

**Mediation between user and data.** A mediator module should accept various data sources without demanding a rigid format or access method. There is richness in diversity. However, a uniform protocol and self-describing data formats are desirable (e.g. Unidata net CDF). Query request would originate by an application and transmitted to the mediator which would interpret the query and plan the data access.

**Social behavior of programs.** A typical data processing application is an integrated and self contained program with processes and user interface "hard wired" (compiled) into the system. Such applications also promote anti-social behavior because they do not play together with other application.

Future data processing programs should not only coexists but cooperate with other applications resulting in user-assembled application that would be much more then the sum of their parts (e.g. Haeberli, 1988)[14,15]. The tools that will facilitate such a new software will include graphic or visual programming languages, data flow structures, module connection managers, etc.

## 3.3 Problems with Flat File Database

Before discussing the design of a modern relational environmental database scheme, we will consider some of the problems with older database designs. A good example is the DATSAV file system for the storage of meteorological data. Its record structure is:

DATSAV (STATION_CODE, ST_LAT, ST_LON, TIME, VAR1, VAR2, .....)

Each of the DATSAV records contain the data for a set of meteorological observations at one station and one time. The problems include:

1 *Redundancy.* The latitude and longitude of the station is repeated for each data record.
2. *Potential inconsistency* (update anomalies). As a consequence of the redundancy, we could update the lat/long for a station in one tuple, while leaving it unchanged in another. Thus, we would not have a unique location for each station.
3. *Insertion anomalies.* We can not record a lat/lon for a station if that station does not currently have at least one data point.
4. *Deletion anomalies.* The inverse to problem (3) is that should we delete all of the data records measured at one location, we unintentionally lose track of the station's lat/lon.

In the DATSAV example, the above problems go away if we replace DATSAV by two relations:

METDATA (STATION_CODE, TIME, VAR1, VAR2, ....)
STATIONS (STATION_CODE, ST_LAT, ST_LON)

Here, STATIONS table, gives the lat/lon for each station just once; hence there is no redundancy. Moreover, we can enter lat/lon for a station even if it currently has no data items.

In the above example, there is a disadvantage to the above decomposition; to find the lat/lon of stations, one must take a relational join (merging of two tables), which is expensive, while with the single relation DATSAV we could simply do a selection and projection.

## 3.4 The Relational Data Model for Environmental Data

In the last few years there have been significant developments on both conceptual and practical levels of database management. The data model that has received a great deal of support is the relational model. For reasons that are not apparent to us, the relational model has not been utilized adequately to organize and access environmental data.

After examining the existing science and technology of database systems we have concluded that for most of the data sets that we are concerned with, such as the aerosol climatic data, the relational model is not only suitable but it has distinct advantages compared to the *ad hoc* structures discused in the previous sections. Some of the characteristics, the advantages and the disadvantages of the relational model are stated below.

In case of the relational model, certain rules and constraints exist that can help mechanizing somewhat the design process. The design of *relation* schemes (set of attributes or table column headings) is still an intuitive process. However, the design of the *database* scheme (collection of relation schemes) follows well defined theory. For this and other reasons we have adopted the relational data model for the organization and manipulation of our environmental data sets.

The central concept in relational database design is data dependency that is the constraint on the construction of relations. For instance, in a location description table the LOC_CODE uniquely determines the LOC_NAME and LOC_LATITUDE. There is a functional dependency of these attributes on LOC_CODE. The concept is somewhat analogous to separation of variables in applied mathematics. All variables that are dependent on a specific variable are grouped to form a relation.

Each relational table has one or a combination of several attributes as its primary key. It provides the means for unique identification of tuples in the relation. Semantically, the primary key is the topic of the relation. The non-key attributes are mere descriptions of the key attribute. Consider the relation LOCATION for example.

LOCATION (LOCATION_CODE, NAME, LAT, LON, ELEVATION

where LOCATION_CODE is the primary key and is a tuple in the relation. The last four values of the tuple describe the primary key.

**Functional Dependencies.** Relations can be used to model the data in several ways; for example, each tuple of a relation could represent an entity and its attributes or it could represent a relationship between entities. One can distinguish two kinds of restrictions on relations:

1. *Restrictions that depend on the semantics of domain elements.* These restrictions depend on the understanding what components of tuples mean. For example, there is no atmospheric temperature of 1000 °C. It is useful to have a DBMS check for such implausible values, which probably arose of an error when entering or computing data. These are simple "integrity constraints".
2. *Restrictions on relations that depend only on the equality or inequality of values.* There are other constraints that do not depend on what value a tuple has in any given component, but only on whether two tuples agree in certain components.

Functional dependencies arise naturally in many ways. For example, if R represents an entity set whose attributes are $A_1,...,A_n$, and X is a set of attributes that forms a key for the entity set, then one may assert X -> Y for any subset Y of the attributes, even a set Y that has attributes in common with X. The reason is that the tuples of each possible relation r represent entities, and entities are identified by the value of attributes in the key. Therefore, two tuples that agree on the attributes in X must represent the same entity and thus be the same tuple.

Similarly, if relation R represents a many-one relationship from entity set $E_1$ to entity set $E_2$, and among the A's are attributes that form a key X for $E_1$ and a key for $E_2$, then X -> Y would hold, and in fact, X functionally determines any set of attributes of R. However, X -> Y would not hold unless the relationship were one to one.

The only way to determine the functional dependencies that hold for the relation scheme R is to consider carefully what the attributes mean. For instance, one simply needs to know that the elevation depends only on the station code, but not on the parameter to be measured. In this sense, dependencies are actually assentations about the real world; they cannot be proved, but we might expect them to be enforced by a DBMS if told to do so by the database designer. Many relational systems enforce those functional dependencies that follow from the fact that a key determines the other attributes of a relation.

### 3.5 Relational Tables for Spatial/Temporal Environmental Data

Following the above guidelines augmented by intuitive arguments, the following relational tables (Table 2) appear reasonable for capturing the meaning of environmental data.

Table 2. Relational tables for spatial/temporal environmental data.

RELATION : Location

| LocCode | StName | Lat | Lon | Elev |
|---|---|---|---|---|
| 103a | Stn1 | 38.35 | 99.55 | 32 |
| 104a | Stn2 | 39.35 | 102.75 | 12 |
| 108b | Stn3 | 38.35 | 108.34 | 22 |
| 109a | Stn4 | 36.35 | 103.67 | 55 |
| 110c | Stn5 | 35.35 | 109.34 | 36 |

RELATION : Parameter

| VarCode | Variable | Unit |
|---|---|---|
| Precip | Precipitation | mm |
| SO4 | Sulfate | umole/l |
| NO3 | Nitrate | umole/l |
| $H^+$ | Hydr. Ion | umole/l |
| Cl | Chloride | umole/l |

RELATION : Time

| Time | Year | Month | Day |
|---|---|---|---|
| 880112 | 88 | 1 | 12 |
| 880113 | 88 | 1 | 13 |
| 880114 | 88 | 1 | 14 |
| 880115 | 88 | 1 | 15 |
| 880116 | 88 | 1 | 16 |

RELATION : Data

| LocCode | VarCode | Time | Value |
|---|---|---|---|
| 103a | Precip | 880112 | 3.84 |
| 103a | Precip | 880113 | 4.69 |
| 103a | SO4 | 880114 | 2.22 |
| 103a | Precip | 880115 | 3.98 |
| 110c | NO3 | 880112 | 2.88 |
| 110c | Precip | 880113 | 1.25 |

# 4. INTERACTIVE DATA EXPLORATION

An implementation of some of the concepts discussed in sections 1-3 of this report is the Voyager data exploration software by Lantern Corporation*. It was designed for browsing and manipulation of large spatial/temporal databases. The specific example used for illustration is the aerosol climate database for Europe.

## 4.1 Browsing of Aerosol Data with the Voyager Software

The Voyager workspace consists of windows each showing a different dimension of the loaded database. The example database shown in Figure 3, EUROBXQ3, consists of the yearly trends (1973-1986) of third quarter extinction coefficients for Europe.

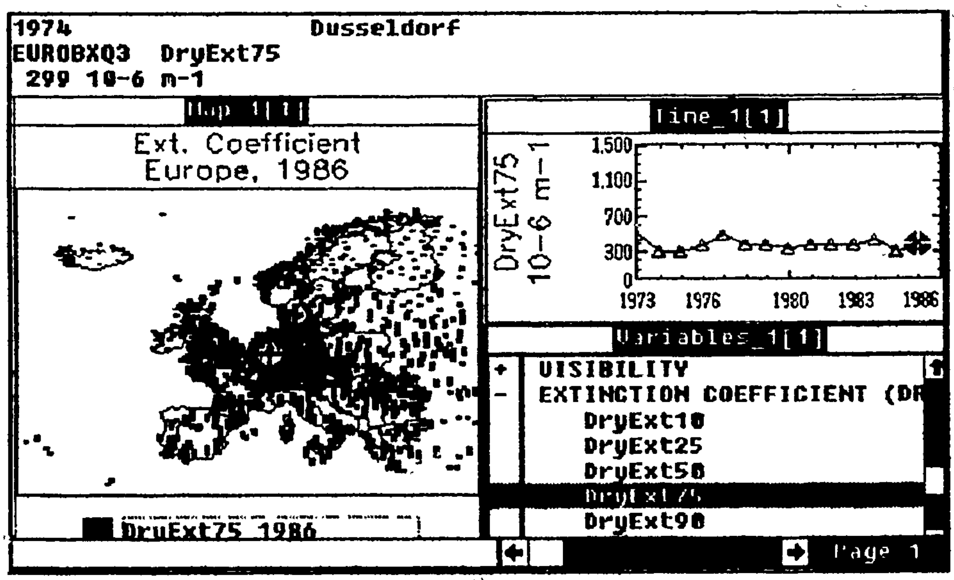


**Figure 3. Voyager's workspace showing the Map, Time, and Variables views of the EUROBXQ3 database.**

**Data Views:** Voyager's workspace is divided into several areas called data views, representing the dimensions of the data space: list of variables, map and time views. The lower right area is the list of variables view. It displays the available

* The principal investigator of this project participated in the design of the Voyager software. He is also an officer of Latern Corporation.

variables in the database. Clicking on a variable such as extinction coefficient (*DryExt75*) selects it as the variable displayed in the other views. The left area is a map view. In this example, the extinction coefficient is displayed as bars for each station. The upper right area is a time view for a particular station (Dusseldorf, FRG). Here the extinction coefficient is displayed as a trend over a selected period of time. Each data view has two purposes, one to display data and second to provide a selection surface for its linked views.

**Linkages and Navigation:** All of these views are informationally linked to each other. Selecting a new time, location, or variable in their respective views simultaneously changes all of the other views.

One can change the current time displayed in the map and variables views by clicking on the time trend line with the mouse. The picture below (Figure 4) shows that the map view is now displaying information corresponding to 1973 instead of 1986. The selection process is similar for the other views and provides Voyager its capacity to create a workbook of living pictures. Do you want to know what the trend looks like for Frankfurt, FRG? Point at Frankfurt, FRG and click the mouse. In general, navigation in the data space is accomplished by pointing and clicking on the set of objects that one whishes to display.

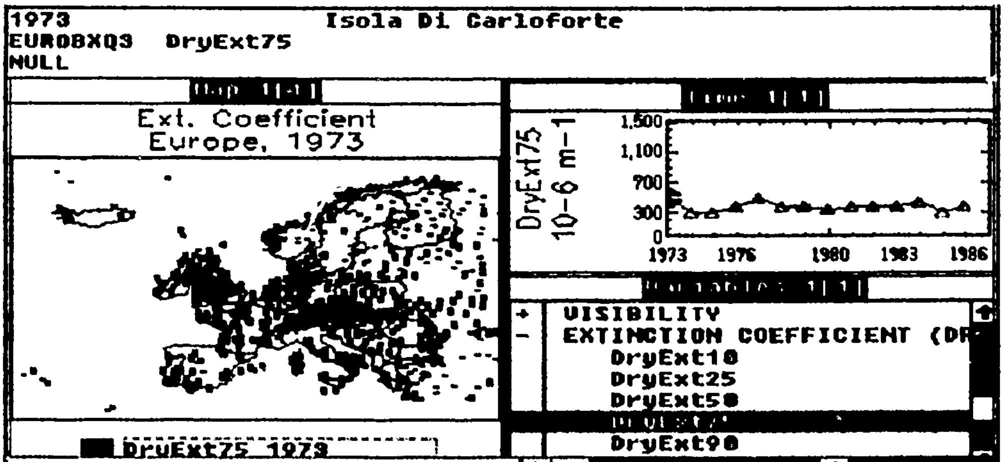


**Figure 4. The data views are linked: changing the cursor position in one view causes an update of the other views. Moving the time cursor from 1986 to 1973 causes an update of the map to show the extinction coefficient for 1973.**

**Zoom in and Zoom out:** Voyager also provides a facility to zoom in on a specific region or time domain. In the example below (Figure 5), the extinction coefficient is shown for Europe in 1986. Do you want to know the extinction coefficient for West Germany? Zoom in map view on West Germany and the map view updates itself to show the selected region, (Figure 6).

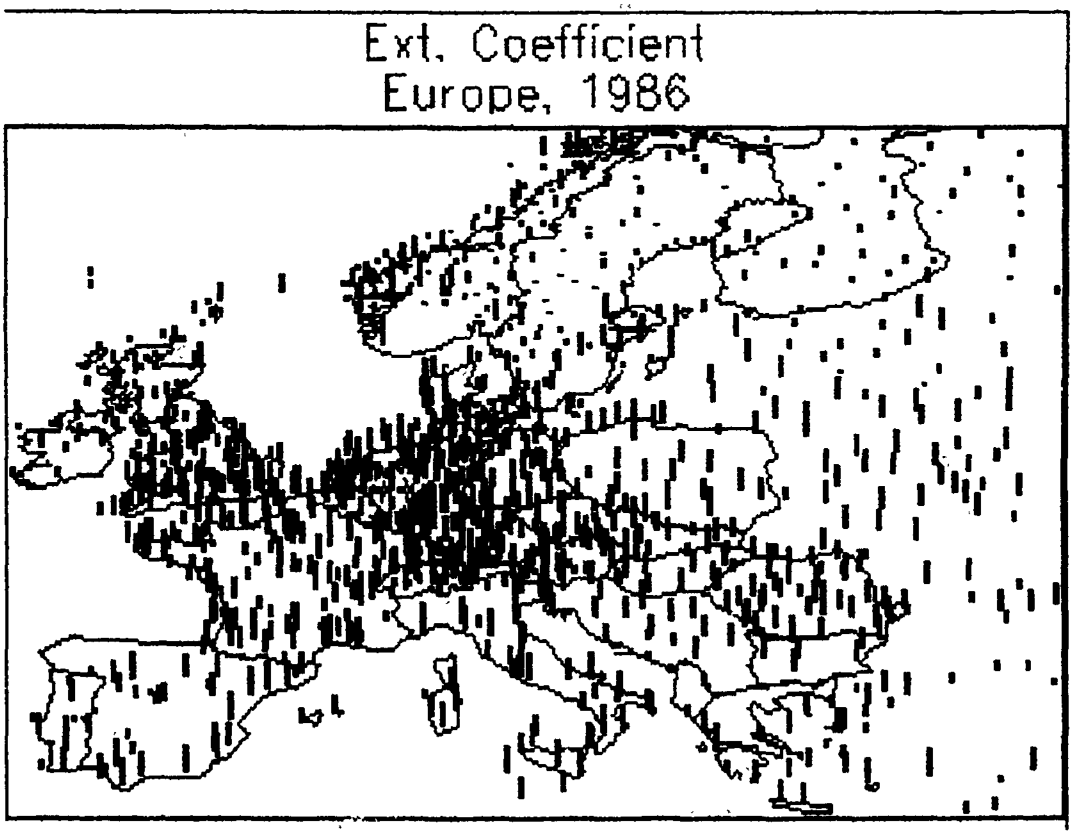


**Figure 5. Extinction coefficient for Europe in 1986.**

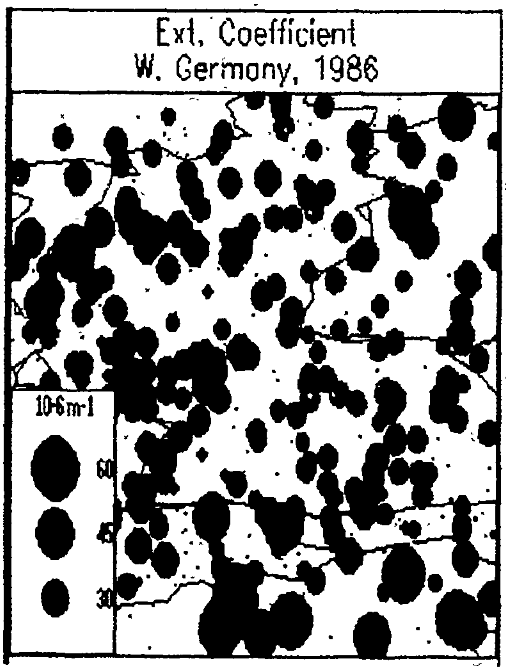


**Figure 6. Extinction coefficient for West Germany for 1986.**

**Overlay of Multiple Data Sets:** Voyager can display more than one database at a time. A time view can contain any number of trends, either from the same database or from another database. The next picture (Figure 7) shows the extinction coefficient for northern Italy. A file structure, with record buffering along with the memory management provided by Microsoft Windows, extends the size of usable databases to the size of the user's disk storage.

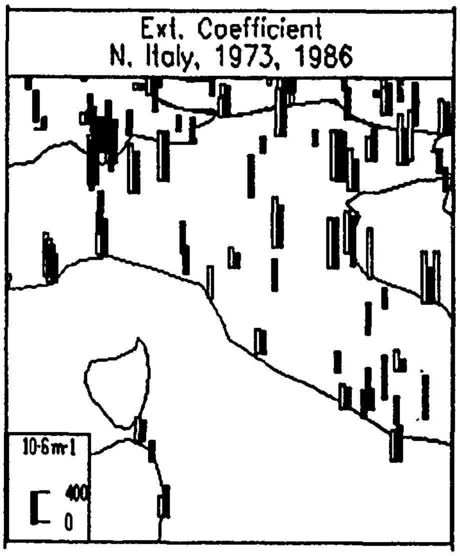


**Figure 7. The map view allows overlaying data for 1973 and 1986 for northern Italy.**

**Graphic Database Query:** The point-and-click selection of display items combined with the linked views is an implementation of a Graphic Query Language (GQL). It is closely related to the Structured Query Language (SQL) used to access data from relational databases. Every click causes the execution of a database command or query compatible with SQL. See Appendix C for more detail.

**Data Manipulation:** Voyager can create new variables from algebraic and logical combinations of existing variables, much like a cell formula in a spreadsheet. This feature helps exploring suspected relationships in various data sets. The graph below (Figure 8), for example, shows the time trend of ratio of extinction coefficients for two stations in West Germany (Frankfurt and Darmstadt).

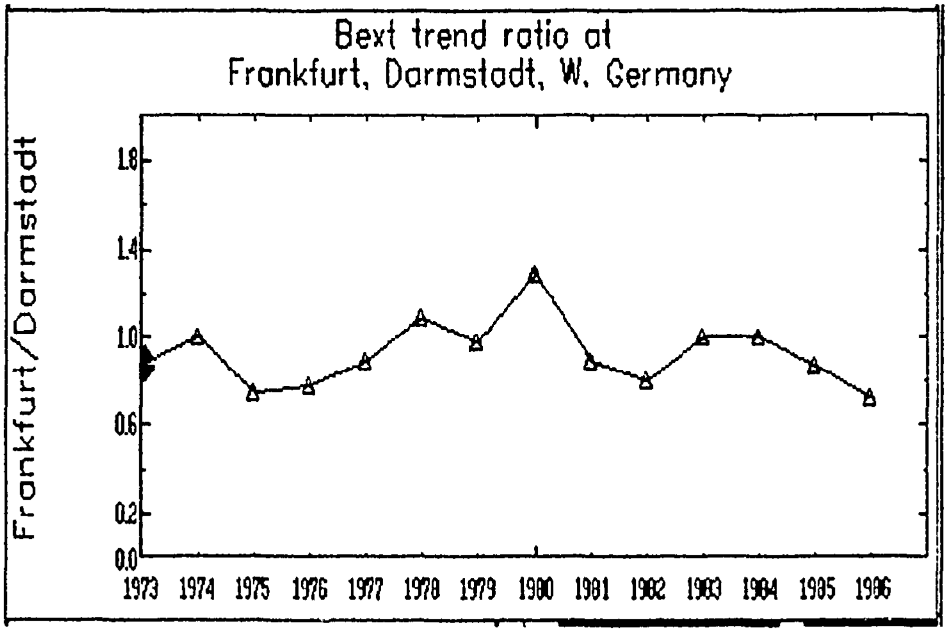


**Figure 8. Virtual variables can be created using the built-in data manipulation language. Here, the ratio of extinction coefficients is computed by dividing extinction coefficient measured at Frankfurt with extinction coefficient measured at Darmstadt.**

**Workbook:** An analogue of the Voyager workspace is a picture book. Each page may contain several figures and text windows. A collection of pages is a dynamic workbook that can be saved and retrieved as a disk file. The author of the workbook may organize and display a story as a pictorial summary report. The workbook contains a portion of the author's knowledge that can be transmitted as a file or presented as a slide show of live pictures. A well authored workbook, along with the hypertext documentation facility turns a pure database into a knowledge base.

Hard copy output from Voyager views can always be produced at the highest quality output that the attached printer can support.

**Database Documentation.** Conventionally, the textual documentation of databases--their source, history, data collection methodology, and other descriptive information--is usually contained in hard copy reports Such documentary knowledge about data is not easily attached to an electronic database. Voyager, however provides a context-sensitive on-line documentation facility for spatial-temporal databases. One can attach text to a database in general or to a specific variable, location, or time.

This on-line documentation is accomplished by a general purpose hypertext Help facility that is dynamically linked to Voyager. The Help facility combines the features of an outliner with a hypertext document browser. For example, pressing the database help key while the cursor is on the parameter extinction coefficient causes the help screen to pop up, displaying the descriptive text about that topic. Similarly, requesting help while pointing at a given location on a map brings up the location-specific text documentation. One can edit the contents of the help text file with a word processor. Hence it can grow as more knowledge and annotations accumulate.

## 4.2 Data Import and Export

High speed browsing and data manipulation is facilitated by a highly indexed binary file format. Prior to usage in Voyager, spatial-temporal data are converted to the internal Voyager format by the Voyager Data Compiler supplied with the software. The compiler accepts as inputs ASCII tables containing the data and data dictionaries. Voyager software can extract subsets of data from compiled Voyager data sets and export those to other programs. Additional translation facilities can be used to and from common data formats, as illustrated in Figure 9.

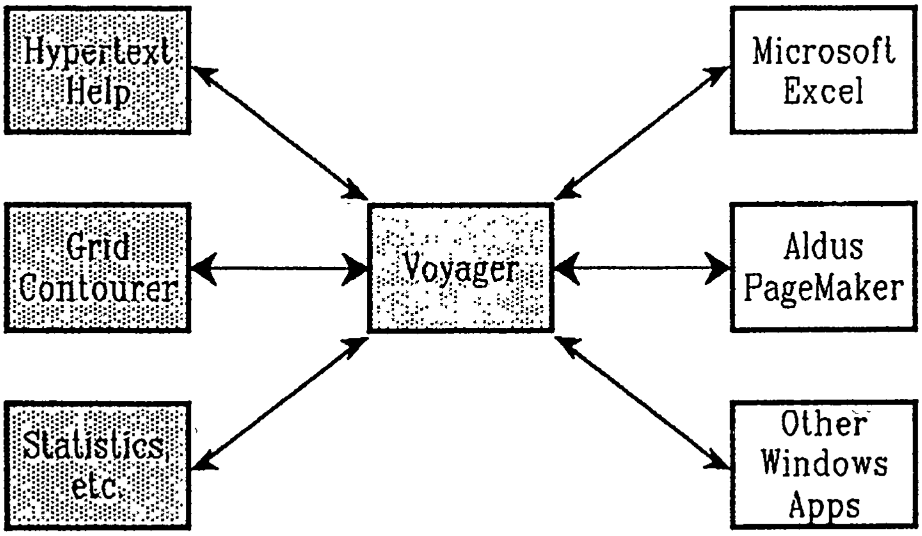


**Figure 9. Voyager can be dynamically linked to cooperating Windows applications providing a seamless linkage.**

# APPENDIX A: DATABASE CONCEPTS

The following is a textbook style presentation of major database concepts. We drew heavily on Ullman, 1988[4].

There are two major characteristics of database management systems (DBMS).

1. They manage *persistent* data, and

2. They *access* large amounts of data efficiently.

The first point states that there is a database which exists of which contents changes slowly. The contents of this database is the data that a DBMS accesses and manages. The second point distinguishes a DBMS from a file system, which also manages persistent data, but does not generally help provide fast access to arbitrary portions of the data. A DBMS 's capabilities are needed most when the amount of data is very large, and the data structure is complex. For small amounts of data, simple storage and access techniques, such as linear scans of the data, are usually adequate. For small data sets the use of spread sheets has provided significant benefits.

Other capabilities that are found in standard DBMS's are:

3) Support for at least one *data model* or mathematical abstraction through which the user can view the data.

4) Support for certain *high-level language* that allow the user to define the structure of data, access data, and manipulate data.

5) *Transaction management,* the capability to provide correct, concurrent access to the database by many users at once.

6) *Access control,* the ability to limit access to data by unauthorized users, and the ability to check the validity of data.

7) *Resiliency,* the ability to recover from system failures without losing data.

Transaction management and access control are less significant for scientific databases while support of high level language is very important.

**Data Models.** Each DBMS provides at least one abstract model of data that allows the user to see information in understandable terms. In fact, it is usually possible to see data at several levels of abstraction. At a relatively low level, a DBMS commonly allows us to visualize data as composed of files.

In many of the data models, a file of records is abstracted to a *relation*, which might be illustrated by:

LOCATION (LOC_CODE, NAME, LATITUDE, LONGITUDE)

Here, LOCATION is the name of the relation. LOC_CODE, NAME, etc are field names; fields are often called attributes in a relation. Relations are often called tables.

A relation is an abstraction of a file, where the data type of fields is generally of little concern, and the order of the records is not specified. Records in a relation are called tuples. Thus, a flat file is a list of records (rows), but a relation is a set of tuples.

**Efficient File Access.** The ability to store a file is not remarkable; the file system associated with any operating system does that. The capability of a DBMS is seen when we access the data of a file. For example, suppose we wish to find the temperature at St. Louis on July 30, 1989. If database has millions of temperature records, it may be very expensive to search the entire file to find the desired data point. A DBMS helps setting up "index files", that allow accessing the specific record in essentially one stroke, no matter how large the file is. Likewise, insertion of new records or deletion of old ones can be accomplished in time that is small and essentially constant, independent of the file's length. DMBS can also help to combine values in two or more files to obtain the information we want.

**Query Language.** To make access to files easier, a DBMS provides a *query language,* or *data manipulation language* to express operations on files. Query languages differ in the level of detail they require of the user, with systems based on the relational data model generally requiring less detail than languages based on other models. A graphic query language discussed in section 4 requires only to point and click on specific objects.

## A.1 Levels of Abstraction in DBMS

Between the computer, dealing with bits, and the ultimate user dealing with concepts such as locations, times, and variables, there are several levels of abstraction. A schematic diagram of the levels of abstraction is shown in Figure A.1, consisting of physical, conceptual, and view levels.

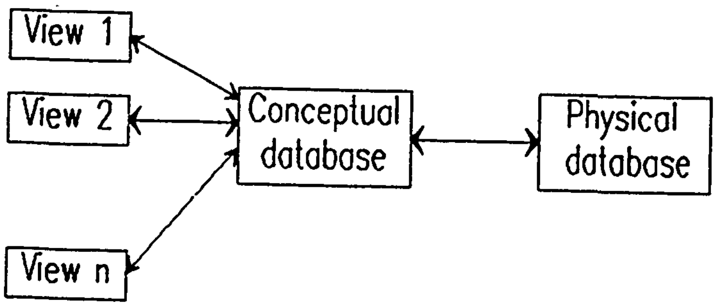


**Figure 10 Levels of database abstraction.**

**The Physical database Level.** A collection of files and indices or other storage structures used to access data efficiently is the *physical database*. The physical database resides permanently on secondary storage devices, such as disks, and many different physical databases can be managed by the same database management system software.

**The Conceptual Database Level.** The *conceptual database* is an abstraction as it pertains to the users of the database. A DBMS provides a *data definition language*, or DDL, to describe the conceptual scheme. The DDL allows describing the conceptual database in terms of a "data model", for example, the relational model. In that model, the data is seen as tables, whose columns are headed by attributes and whose rows are "tuples,". Another example of a suitable data model is a directed graph, where nodes represent files or relations, and the arcs from node to node represent associations between two such files.

The conceptual database is intended to unify all or most of the data used by an organization. The advent of database management systems allows bringing all its files of data together and to see and manipulate them in one consistent way - the way described by the conceptual database. This bringing together of environmental data has not taken place in most organizations. Information of the same type are still typically kept in different places, and the format used for the same kind of information may frequently be very different.

**The View Level.** A *view* or *subscheme* is a portion of the conceptual database. A DBMS provides a facility for declaring views. and a facility for expressing queries and operations on the views. For example, a view to a subset of an environmental database is a map showing the values for a given region.

## A.2 Data Independence

The chain of abstractions of Figure A.1 from view of conceptual to physical database, provides two levels of "data independence". In a well-designed database system the physical scheme can be changed without altering the conceptual scheme or requiring a redefinition of subschemes. This independence is the *physical data independence*. It implies that modifications to the physical database organization may affect the efficiency of application programs, but it will not be required that one rewrites these programs because the physical scheme has changed. The value of physical data independence that it allows "tuning" of the physical database for efficiency while the application programs are unchanged.

The relationship between views and the conceptual database also provides a type of independence called *logical data independence*. As the database is used, it may become necessary to modify the conceptual scheme, for example, by adding information about different types of entities or extra information about existing ones. Modifications to the conceptual scheme can be made without affecting existing subschemes. Enforcing logical data independence is more difficult than ensuring physical independence.

## APPENDIX B: RELATIONAL DATA MODEL

Since its formulation (Codd, 1970)[16], the relational model has had a major impact on the management of data. Unlike earlier data models, i.e., the hierarchical and the network models, the relational model can be defined in terms of simple yet rigorous mathematical concepts. Further, its underlying structure lends itself to easy querying, modification, updating, and database restructuring.

In this appendix, we review briefly the relational model. This review closely follows Parsaye and Chignell (1988)[2]. We also show queries with query language SQL that may be performed on relational databases.

**The relational data Model.** A relational database consists of a set of tables. Each table consists of columns (also referred to as attributes or fields) and rows of data records. A row in a table represents a relationship among a set of attributes. Note that location code is used to identify the location uniquely in this table. In each table there will generally be one attribute or combination of attributes that is referred to as the index or primary key. For instance, Social Security numbers are used to distinguish people since names cannot be guaranteed to be unique.

We will refer to each row in a table as a record or tuple. In relational databases, each record usually describes a single data object. The attributes that describe records (i.e., the columns in the table) are referred to as fields. The fields of the table shown above are LOC_CODE, NAME, LATITUDE, LONGITUDE.

Each field has a domain, or set of possible values; for example, latitude is a floating point number between $+\pi/2$ and $-\pi/2$. Domains may be defined less strictly, as a variable of a certain predefined data type, for example, we could define LOC_CODE as an alphanumeric character string, with no specification of upper and lower bounds.

The definition of a table of fields and domains is called the database schema. The particular table shown above, and many others of the same type, are called database instances. For most purposes, the schema of a database is all a database designer may be concerned with from a logical point of view. The concept of a relation schema may seem similar to a form template with slots.

**Operating on Relations.** The relational algebra is a method for operating on relations. The five fundamental operations in the relational algebra are:

Selection.
Projection.
Product.
Union.
Set difference.

In addition, the join is another operation that may be defined in terms of the cartesian product and selection operations.

The selection and projection operations are unary operations, since they operate on one relation. The other three relations operate on pairs of relations and are, therefore, called binary operations. Joins can be performed on two or more tables.

**Selection.** Selection involves taking a table and selecting the row that satisfy a predicate. For example, you can select all locations with latitude $< 35°$ from the location table. The selection operator takes a table and predicate (in this case, Latitude $< 35°$) as input and returns another table as output.

**Projection.** Projection removes certain columns from a table. For example, you might want to see just the station name and elevation fields from the Location table. The projection operator takes a table and a set of field names as input and returns another table as output. The resulting table contains the same number of rows but fewer columns.

**Product.** The third basic operator is the product. The product multiplies two tables so that if one table has N rows and the other has M rows, the product table will contain (N*M) rows.

**Union and Difference.** The union of two tables with N and M rows, respectively, is obtained by concatenating them into one table with a total of (N+M) rows. Union will generally make sense only if the schemes of the two tables match, i.e., if they have the same number of fields with matching attributes in each field.

The difference operator can be used to find records that are in one relation, but not in another. The difference between A and B is often denoted by A - B.

Additional operators can be defined to simplify the process of building queries, but the essential elements of the relational algebra consist of the operators defined above along with intersection, which can be expressed by using differences and unions. The intersection between two relations A and B can be expressed as

$$A - (A - B)$$

and contains all records in A that are also in B.

**Joining Tables.** The join operator allows us to join several tables together. Although the join is not one of the basic five relational operators, it can often be very useful.

Given two tables, A and B, the join operator allows one to apply a predicate to all rows that are formed by concatenating rows from A and rows from B. Those rows satisfying the predicate are returned. For example, if we have 3 rows in A and 7 rows in B, we can form 21 possible rows to apply the predicate to (this is simply the product of the two tables). Only some of the rows have any meaning.

## APPENDIX C: STRUCTURED QUERY LANGUAGE

The relational algebra provides a concise language for representing queries. However, database system products require query languages that may be easily used and interfaced to programs. Two major approaches are based on the languages SQL (Chamberlin, et al. 1976)[17] and QUEL (Stonebraker, et al. 1976)[18]. In this section we review the language SQL (Structured Query Language), which is the most widely used relational query language.

SQL is, in fact, more than merely a database query language. It includes features for defining the structure of the data, modifying the data, and specifying security constraints within the database.

SQL expressions are made up of three clauses: select, from, and where. A query in SQL has the form

ELECT Attribute1, Attribute2, . . . ,Attribute
FROM Relation1, Relation2, . . . , Relation
WHERE Predicate

In this notation, the list of attributes may be replaced with a star (*) to select all attributes of all relations appearing in the from clause.

The SELECT clause in SQL (confusingly) corresponds to projection operation described above for the relational algebra. In SQL, a SELECT is used to list the attributes desired in the table produced by a query. Finally, the WHERE clause corresponds to the selection predicate of the relational algebra. If the WHERE clause is omitted, no selection is made. If the WHERE clause is present it consists of a predicate involving attributes of the relations that appear in the FROM clause.

The result of an SQL query is a relation. SQL acts as follows:

a. It forms the product of the relations named in the from clause.
b. It performs a relational selection using the where clause predicate.
c. It projects the result onto the attributes of the select (in the SQL sense) clause.

Some features of SQL are not available in the relational algebra. For instance, aggregate operators such as avg (average) operate on aggregates of records, providing statistical and related information, while sorting procedures can be used to order records according to their attributes. The flexibility of SQL can be further enhanced linking it with general-purpose languages such as Pascal and C. This allows the development of customized routines to manipulate the database and makes the SQL language extendible.

The query "Find the station names, latitude, longitude, of all stations in the Location table" can be expressed in SQL as

```
SELECT          StName, Lat, Lon
FROM            Location
```

The result of this query will be a list of all station names, lat/lon in the table.

```
SELECT          StName, Lat, Lon,Value
FROM            Data, Location,Variable
WHERE           Variable="BEXT" and
                Time="880112"
                Latitude<40°
ORDER BY Lat    Latitude<35°
```

It is possible to combine SQL statements into fairly complex queries. SQL statements may also be embedded in programming languages for general-purpose computation.

**Summary.** The relational data model provides a simple yet rigorous model of viewing data. The basic concept underlying the relational model is a table template with a set of fields, called a schema. Each schema may have a set of records, which define a table. By use of the relational algebra, tables may be joined, projected, selected from, etc.

The language SQL provides a formal yet intuitively easy way of dealing with queries. It may be used to select items from tables based on characteristics expressed as field values. When combined with a graphic pre- and post-processor, then SQL is a powerful, simple to use query language.

## REFERENCES


1. USAFETAC DATSAV Database Handbook (1977) Unites States Air Force, Air Weather Service, ASAF Environmental Technical Applications Center, Scott Air Force Base, IL 62225, USAFETAC-TN-77-2.

2. Parsaye, K. and Chignell, M. (1988) Expert Systems for Experts, John Wiley & Sons, Inc., New York.

3. Kerschberg, L. (1987) Proceedings from the First Conference on Expert Database Systems, The Benjamin/Cummings Publishing Co., Inc., Menlo Park.

4. Ulmann, J.D. Principles of Database and Knowledge Base Systems, Volume I, Computer Science Press, Rockville, MD.

5. Brodie, M.L. and Mylopoulus J. Eds. (1986) On Knowledge Base Management Systems, Springer Verlag, New York.

6. Parsaye, K., Chignell, M., Khoshafian, S., and Wong H. (1989) Intelligent Databases Object-Oriented Deductive Hypermedia Technologies, John Wiley & Sons, Inc., New York.

7. McPherron, R. L. (1976) A self-documenting source-independent data format for computer processing of tensor time series, Phys. Earth Planet. Inter., 12, 103-111.

8. Barnes, S. L. (1980) Report on a meeting to establish a common Doppler radar exchange format, Bull. Amer. Meteor. Soc., 61, 1401-1404.

9. Smith, A. Q., and Clauer, C. R. (1986) A versatile source-independent system for digital data management, EOS, Trans. AGU, 67, 188.

10. Mohr, C. G., Miller, L. J., Vaughan, R. L., and Frank, H. W. (1986) The merger of mesoscale data sets into a common Cartesian format for efficient and systemic analyses, J. Atmos. Oceanic Technol., 3, 143-161.

11. Treinish, L.A., and Gough, M.L. (1987) A software package for the data-independent management of multidimensional data, EOS, Trans. AGU, 68, 633-635.

12. Raymond, D.J. (1988) A C language-based modular system for analyzing and displaying gridded numerical data, J. of Atmos. and Oceanic Technol., 5, 501-511.

13. Rew, R.K. (1988) netCDF User's Guide, An Interface for Data Access, Beta Version, Unidata UCAR, Boulder, CO.

14. Haeberli, P. E. (1986) A Data-Flow Manager for an Interactive Programming Environment, Proceedings of Usenix Summer Conference, 1986.

15. Haeberli, P. E. (1988) ConMan: A Visual Programming Language for Interactive Graphics, Computer Graphics, Vol 22, 4, 103-111.

16. Codd, E. F. (1970) A relational model of data for large shared data banks, Commun. ACM 13, 377-387.

17. Chamberlin, D. D., Astrahan, M. M., Eswaran, K. P., Griffits, P. P., Lorrie, R. A., Mehl, J. W., Reisner, P., and Wade, B. W. (1976) SEQUEL 2: A unified approach to data definition, manipulation, and control, IBM Journal of Research and Development, 20, 560-575.

18. Stonebraker, M., Wang, E., Kreps, P., and Held, G. (1976) The Design and Implementation of INGRES, J. of ACM Transactions on Database Systems, 1, 189-222.